(14

เทคนิคการออกแบบโปรแกรม ตอน 2

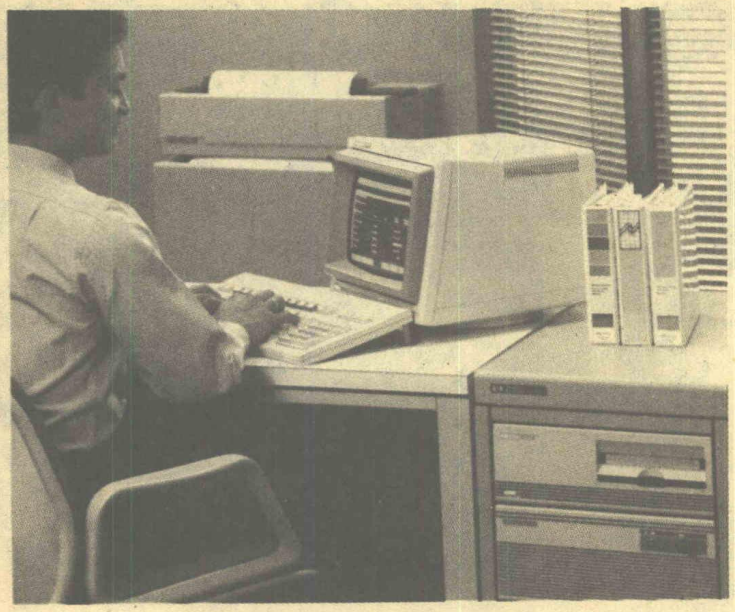
การออกแบบโปรแกรมสมัยใหม่

อาตาปี & VIP

หลักการออกแบบโปรแกรมที่เราจะได้ศึกษาในชั้นต้นนี้ ความจริงไม่ใช่หลักการใหม่ทั้งหมด เพราะมีผู้คิดค้นเอาไว้นานมาแล้ว แต่ที่ใช้คำว่าสมัยใหม่ก็เพราะยังเป็นหลักการที่ใช้กันอยู่ทั่วไปในปัจจุบันนี้ สำหรับในตอนนี้เราจะได้พิจารณาเฉพาะหลักการกว้างๆเป็นการปูพื้นไปสู่รายละเอียดที่จะเสนอต่อไป อนึ่ง เทคนิคที่นำมาอธิบายในที่นี้เป็นเพียงส่วนหนึ่งของเทคนิคการออกแบบโปรแกรมที่มีใช้อยู่เท่านั้น ในวงวิชาการวิศวกรรมซอฟต์แวร์ยังมีเทคนิคอื่นๆ อีกมาก ซึ่งผู้สนใจอาจติดตามศึกษาได้จากตำราเล่มอื่นๆ

ข้อที่พึงระลึกก็คือ วิธีการออกแบบโปรแกรมสมัยใหม่นี้ช่วยทำให้นักโปรแกรมผลิตโปรแกรมที่ดีได้ก็จริง แต่มันก็ไม่ใช่หลักการที่สามารถทำให้นักโปรแกรมทุกคนเป็นผู้ที่มีความสามารถยอดเยี่ยมในด้านการเขียนโปรแกรมได้ เช่น ไม่สามารถทำให้นักโปรแกรมที่ไร้ความสามารถให้กลับเป็นผู้ที่มีความสามารถได้ แต่โดยทั่วไปสามารถช่วยให้นักโปรแกรมทั่วไปปรับปรุงโปรแกรมให้ดีขึ้น ช่วยให้นักโปรแกรมที่ขาดประสบการณ์ให้เรียนรู้ได้เร็วขึ้น และเป็นการช่วยให้การพัฒนาซอฟต์แวร์รวดเร็วตามไปด้วย

หลักการออกแบบโปรแกรมสมัยใหม่นั้นมีส่วนที่แตกต่างกันบ้าง และคล้ายกันบ้าง ไม่มีวิธีใดเลยที่ผิดแผกแตกต่างจากวิธีอื่นอย่างสิ้นเชิง การออกแบบโปรแกรมนั้นเราจำเป็นจะต้องมีขั้นตอนและมีวินัยในการออกแบบ แต่การยึดมั่นในขั้นตอนมากเกินไปก็เป็น



ดาบสองคม เพราะเท่ากับปิดกั้นไม่ให้นักออกแบบโปรแกรมแสดงความคิดสร้างสรรค์ได้ ดังนั้นแม้ว่าเราควรจะต้องยึดมั่นอยู่กับขั้นตอนโดยเฉพาะของวิธีการต่างๆ แต่ก็ควรมีความยืดหยุ่นในการคิดออกแบบพอสมควร

หลักการแบบจากบนลงล่างและจากล่างขึ้นบน

หลักการออกแบบโปรแกรมแบบแรกที่จะพิจารณาคือ หลักการแบบจากบนลงล่าง (Top-down) และจากล่างขึ้นบน (Bottom-up)

การออกแบบชนิดจากบนลงล่างนั้น เราหมายถึงวิธีการย่อยปัญหาโปรแกรมโดยเน้นที่โครงสร้างควบคุมของโปรแกรม หรือการควบคุมการทำงานตามคำสั่งต่างๆในโปรแกรมนั่นเอง ต่อจากนั้นเมื่อได้โครงสร้างควบคุมแล้วจึงจะพิจารณารายละเอียดของแต่ละคำสั่ง งานขั้นแรกของการออกแบบชนิดนี้คือการศึกษาปัญหาที่จะต้องเขียนโปรแกรมให้ถี่ถ้วนแล้วย่อยปัญหานั้นลงเป็นส่วนย่อยๆ หรือโมดูลที่เป็นอิสระต่อกันสัก 3 ถึง 10 ส่วน งานขั้นที่สองก็คือพิจารณาแต่ละโมดูลแล้วย่อยต่อไปอีกเป็นโมดูลย่อยที่เป็นอิสระต่อกันต่อไป เราจะย่อยโมดูลย่อยนี้ต่อไปเรื่อยๆ จนกระทั่งเราได้โมดูลที่มีขนาดเล็กพอที่จะเข้าใจรายละเอียดได้ถี่ถ้วน และสามารถเขียนเป็นคำสั่งที่ตรงไปตรงมาไม่ซับซ้อนได้ในเวลาเดียว โดยปกติเราไม่ต้องย่อยทุกโมดูลให้เป็นโมดูลย่อยๆเหมือนกันหมด โมดูลบางอันอาจต้องย่อยต่อ แต่โมดูลบางอันอาจจะไม่จำเป็นต้องย่อยต่อไปอีกก็ได้

ลักษณะสำคัญอย่างหนึ่งของการออกแบบชนิดจากบนลงล่างก็คือ ขณะที่กำลังพิจารณาโมดูลแต่ละระดับนั้น เราไม่จำเป็นต้องสนใจรายละเอียดของโมดูลในระดับล่างที่ย่อยลงไป เราเพียงแต่สนใจเฉพาะข้อมูลและการควบคุมการทำงานของโมดูลระดับนั้นเท่านั้น นี่หมายความว่าเราจะต้องกำหนดข้อมูลที่จะส่งผ่านระหว่างโมดูลต่างๆให้ครบถ้วน แต่เราไม่จำเป็นจะต้องระบุโครงสร้างข้อมูลที่อยู่ในโมดูลระดับล่างจนกว่าเราจะพิจารณาลงไปถึงระดับนั้นแล้ว อย่างไรก็ตามถ้าหากมีการใช้ข้อมูลร่วมกันในโมดูลต่างๆ ที่อยู่ในระดับเดียวกันแล้ว เราจำเป็นจะต้องเลือกโครงสร้างข้อมูลนั้นให้เสร็จก่อนจะพิจารณารายละเอียดในระดับล่าง และออกแบบรายละเอียดของแต่ละโมดูลนั้น เราจำเป็นต้องพิจารณาโครงสร้างข้อมูล และวิธีการอ่านข้อมูลในโมดูลนั้นให้เสร็จ

นอกจากการออกแบบชนิดจากบนลงล่างแล้ว เราอาจจะเขียนโปรแกรมในแบบจากบนลงล่างได้ โดยเราจะเริ่มจากรายละเอียดของโครงสร้างควบคุมที่เราเริ่มย่อยปัญหาในระดับแรกแล้วเขียนคำสั่งต่างๆที่เกี่ยวข้องกับโครงสร้างควบคุมนั้น ต่อจากนั้นก็เริ่มเขียนคำสั่งสำหรับโครงสร้างควบคุมในระดับย่อยลงไปทีละระดับ จนกระทั่งถึงคำสั่งในโมดูลระดับล่างที่สุด

คำถามที่น่าสนใจก็คือ เราควรจะย่อยโมดูลให้ละเอียดลงไปถึงขนาดไหน นักคอมพิวเตอร์บางคนมีความเห็นว่าควรมีขนาดหนึ่งหน้ากระดาษ หรือประมาณ 50 บรรทัด แต่บางคนเช่น ฮารลาน มิลล์ เห็นว่า โมดูลควรจะมีขนาดใหญ่เท่าที่นักโปรแกรมจะมีความสามารถคิดพิจารณาได้ ซึ่งในกรณีนี้หมายความว่านักโปรแกรมสามารถวางกระดาษพิมพ์โปรแกรมกระจายไว้บนโต๊ะ แล้วนั่งอ่านนั่งคิดพิจารณาคำสั่งทั้งหมดได้ โดยไม่ต้องย่อยลงไปเป็นโมดูลย่อยอีก

ในกรณีของการออกแบบชนิดจากล่างขึ้นบนนั้น นักออกแบบจะพิจารณาปัญหาที่มีแล้วใช้ประสบการณ์หรือแรงดลใจตัดสินว่า ส่วนใดในปัญหาเป็นส่วนที่ยุ่งยากหรือซับซ้อนมากที่สุด เมื่อได้แล้วก็จะทุ่มเทความสนใจให้กับส่วนนี้ โดยตัดสินใจเกี่ยวกับรายละเอียดและโครงสร้างของโปรแกรมส่วนนี้จนเสร็จแล้วจึงค่อยพิจารณาปรับปรุงรายละเอียดส่วนอื่นให้เข้ากับรายละเอียดของส่วนแรก

การออกแบบชนิดจากล่างขึ้นบนนี้มีส่วนคล้ายคลึงกับการออกแบบโดย

การสังเคราะห์ที่กล่าวไปแล้วอยู่บ้าง โดยที่ข้อกำหนดความต้องการของโปรแกรมจะเน้นที่ส่วนที่สำคัญที่สุด แล้วการออกแบบก็เริ่มจากจุดที่สำคัญที่สุดนี้ ถ้าหากออกแบบส่วนที่สำคัญที่สุดนี้ตามข้อกำหนดไม่ได้ ก็ต้องเปลี่ยนแปลงข้อกำหนดใหม่ก่อนที่จะเริ่มออกแบบอีกครั้ง

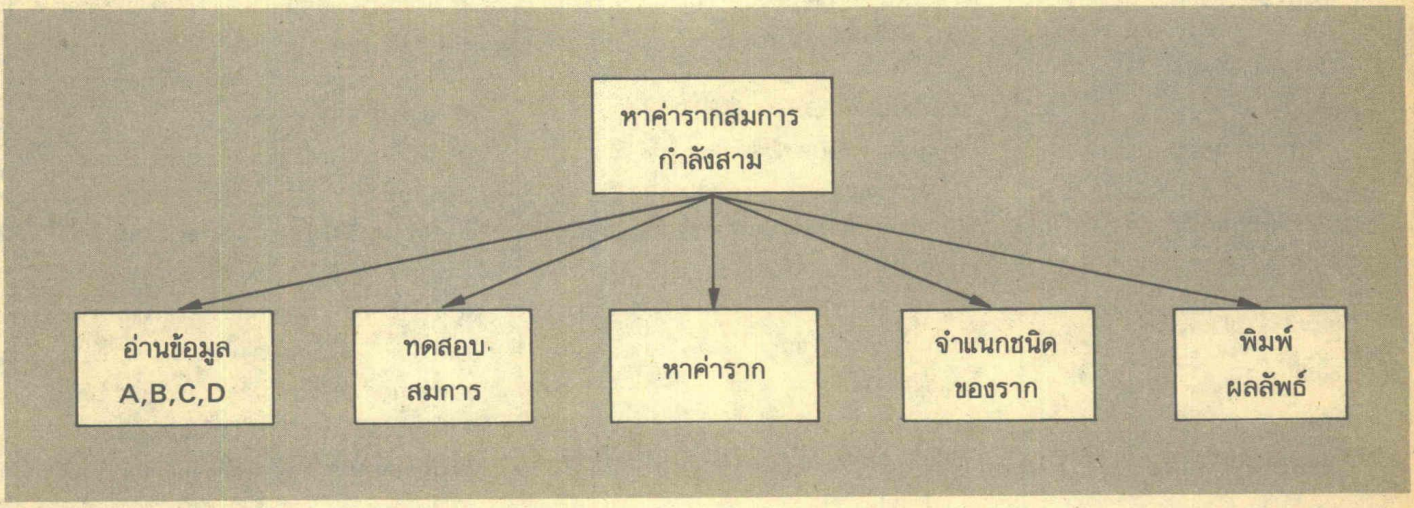
จุดเด่นของการออกแบบชนิดจากบนลงล่างอยู่ที่เราไม่ต้องพะวงถึงรายละเอียดในตัวโปรแกรมจนกระทั่งถึงขั้นสุดท้ายของการออกแบบ ดังนั้นในระหว่างการออกแบบนั้นถ้าหากมีการเปลี่ยนแปลงเล็กๆน้อยๆในข้อกำหนดหรือในเทคโนโลยีที่จะเอามาใช้ก็ยังไม่เป็นไร แต่จุดอ่อนสำคัญก็คือถ้าหากว่าข้อกำหนดนั้นทำตามไม่ได้ และเราไม่รู้ว่าทำไม่ได้ จนกระทั่งถึงขั้นสุดท้ายแล้วก็เท่ากับทำให้เราเสียเวลาไปโดยเปล่าประโยชน์ ในทางตรงกันข้ามการออกแบบชนิดจากล่างขึ้นบนนั้นจะพิจารณาส่วนที่สำคัญที่สุดเป็นประเดิม ดังนั้นถ้าหากข้อกำหนดนั้นทำตามไม่ได้นักออกแบบก็จะรู้ก่อนในขั้นแรก อย่างไรก็ดี โครงสร้างควบคุมของโปรแกรมที่สร้างขึ้นให้สัมพันธ์กับส่วนที่สำคัญที่สุดนั้นอาจจะไม่ดีพอก็ได้

ในทางปฏิบัติจริงๆ เราอาจจะออกแบบในลักษณะจากบนลงล่าง และเขียนโปรแกรมในลักษณะจากล่างขึ้นบนก็ได้ และเราประสบปัญหาที่ซับซ้อนเราอาจจะต้องใช้เทคนิคทั้งสองอย่างนี้ร่วมกันไป โดยเริ่มจากตรงกลาง แล้วออกแบบจากจุดนั้นลงล่างและขึ้นบน

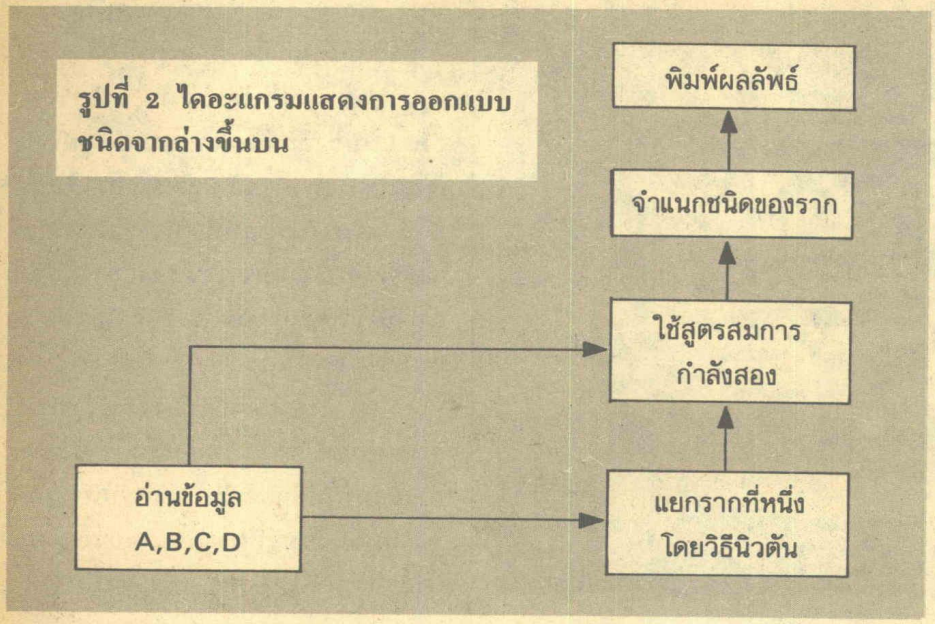
เพื่อให้เห็นตัวอย่างระหว่างการออกแบบชนิดจากบนลงล่าง และจากล่างขึ้นบนให้ชัดเจนยิ่งขึ้น ขอให้พิจารณาโปรแกรมสำหรับหาค่ารากของสมการกำลังสาม $Ax^3+Bx^2+Cx+D=0$

เมื่อใช้เทคนิคการออกแบบชนิดจากบนลงล่าง เราจะแบ่งปัญหานี้ออกเป็นห้าโมดูลด้วยกัน โมดูลแรกมีหน้าที่อ่านข้อมูล A, B, C และ D โมดูลที่สองใช้ทดสอบว่าสมการนี้จะหาค่ารากได้หรือไม่ โมดูลที่สามเป็นส่วนที่ใช้หาค่าราก โมดูลที่สี่ทำหน้าที่แยกคำตอบว่าเป็นค่ารากชนิดใด และโมดูลสุดท้ายใช้พิมพ์ค่ารากเป็นผลลัพธ์ ดังรูปที่ 1

แต่ถ้าใช้เทคนิคการออกแบบชนิดจากล่างขึ้นบน เราจะเริ่มนึกถึงว่าเราจะหารากได้อย่างไร เราอาจจะรู้ว่าวิธีหาค่ารากคำแรกโดยวิธีของนิวตัน แล้วใช้



รูปที่ 1 ไดอะแกรมแสดงการออกแบบชนิดจากบนลงล่าง



รูปที่ 2 ไดอะแกรมแสดงการออกแบบชนิดจากล่างขึ้นบน

วิธีหาค่ารากของสมการกำลังสองเป็นขั้นต่อไป ดังนั้น จุดนี้จึงเป็นส่วนสำคัญที่เรากำหนดรายละเอียดขึ้นก่อน แล้วจึงสร้างรายละเอียดอื่นๆตามมา ดังรูปที่ 2

ขอให้สังเกตว่า ถ้าหากเราต้องการเปลี่ยนวิธีหาค่าราก เราจะต้องเปลี่ยนโครงสร้างของโปรแกรมในรูปที่ 2 ใหม่ทั้งหมด แต่ในกรณีของการออกแบบจากบนลงล่าง เราไม่ต้องเปลี่ยนโครงสร้างใหม่ เราเพียงแต่เปลี่ยนโมดูลที่ใช้หาค่ารากเท่านั้น

ความแตกต่างระหว่างการออกแบบทั้งสองวิธีนี้ยังมีอีกหลายข้อ ดังได้นำมาสรุปไว้ในตารางที่ 1

ตารางที่ 1 ความแตกต่างระหว่างการเขียนโปรแกรมจากบนลงล่าง และจากล่างขึ้นบน

ลักษณะ	บน-ลง-ล่าง	ล่าง-ขึ้น-บน
1. ข้อกำหนด (Specification)	ต้องการข้อกำหนดที่รัดกุมรวมทั้งรายละเอียดของสิ่งแวดล้อมที่ระบบจะทำงาน	อาจเริ่มงานได้จากข้อกำหนดของโมดูลส่วนที่สำคัญที่สุดเท่านั้น
2. ข้อผิดพลาดในการออกแบบ	ถ้าทำผิดในระดับบนและไปรู้ทีหลังก็จำเป็นจะต้องออกแบบกันใหม่หมด แต่ถ้าผิดในระดับล่างก็ไม่ต้องแก้ไขมากนัก	ความผิดพลาดในระยะแรกไม่ค่อยจะมีผลกระทบมากนัก เพียงแต่ทำให้ต้องออกแบบและเขียนโปรแกรมเฉพาะแต่ละโมดูลใหม่เท่านั้น
3. การทดสอบ	ต้องใช้โมดูลสมมติในการทดสอบ (test stub)	ต้องเขียนโปรแกรมขึ้นเรียกโมดูลที่จะทดสอบ (Test driver)
4. มาตรฐาน	สามารถสร้างโมดูลหรือโครงสร้างมาตรฐานเอาไว้ใช้งานต่อไปได้	สามารถสร้างโมดูลมาตรฐานเอาไว้ใช้งานต่อไปได้
5. โครงสร้างข้อมูล	ยังไม่ต้องกำหนดโครงสร้างข้อมูลโดยละเอียดในระหว่างการออกแบบขั้นต้น	จำเป็นจะต้องกำหนดโครงสร้างข้อมูล พร้อมทั้ง interface ให้เสร็จก่อน
6. ขนาดของโมดูล	ประมาณ 1 หน้า หรือ 50 บรรทัด ถ้ายาวกว่านี้ให้ย่อยลงไป	ขนาดขึ้นอยู่กับฟังก์ชันของงานอาจจะยาวหรือสั้นก็ได้
7. การทดสอบรวม	ไม่มีปัญหาเพราะจะเคยทดสอบ interface ระหว่างโมดูลมาแล้ว	บางครั้งอาจมีปัญหา โดยเฉพาะทางด้าน interface
8. ความเหมาะสม	ถ้าเคยศึกษาความเหมาะสมของงานเขียนโปรแกรมชิ้นนี้มาแล้ว ก็สามารถใช้วิธีจากบนลงล่างได้เลย	ถ้ายังไม่เคยศึกษาความเหมาะสมของงานมาก่อนควรใช้วิธีจากล่างขึ้นบนกับงานส่วนสำคัญเพื่อพิสูจน์ว่าทำได้จริงก่อน จากนั้นจึงค่อยใช้วิธีจากบนลงล่างทำส่วนที่เหลือ
9. การเพิ่มเติมโมดูล	ทำได้ง่าย และอาจจะออกแบบเผื่อเอาไว้ล่วงหน้าได้ด้วย	อาจมีปัญหามาก จนถึงกับต้องออกแบบใหม่หมด ถ้าหาก interface เข้ากันไม่ได้